Improving Medicine Package Product Quality Control using Image Recognition Machine Learning

Ritam Chakraborty

Received October 26, 2023
Accepted July 03, 2024
Electronic access July 15, 2024

Using machine learning to optimize the quality control of packaging is crucial to cutting down manufacturing costs and time delays. At this moment there has been a lack of direct research on using image recognition to find defects in medicine packaging specifically. This research explores various methods for using image classification to determine whether a medicine package is intact or damaged using a side view and a top view of the package. The images of the packages are first preprocessed using an edge detection algorithm. Two separate machine learning algorithms, one for the top and side view, are then used to classify whether the package is intact or damaged. If either the model trained on the top view or the side view determines the package is damaged, our algorithm concludes that the package is indeed damaged, and shouldn't be shipped to consumers. If both models determine the package is intact, the algorithm concludes that the package is indeed intact. After preprocessing the images with an edge detection algorithm, we train several models (Support Vector Machine, Random Forest Classifier, Naive Bayes Classifier, and K-nearest Neighbors) to find the defective images in the dataset.

Introduction

Industrial package damage has proven to be a persistent issue for many businesses on both large and small scales. Up to 11% of packages arriving at distribution centers have some level of damage and 34% of returns related to packaging are a result of packaging damage[1,2]. For these companies, these defects result in lost revenue, lower brand reputation, and greater customer dissatisfaction. Though a significant proportion of packaging damage can be attributed to transit, the manufacturing process can also be adjusted to ship fewer damaged packages.

Traditionally, workers have been employed to analyze packaging for damages. However, machine learning, specifically image recognition, can effectively and swiftly detect irregularities on packages, eliminating the need for human involvement. Despite the potential subtlety of damages on specific types of packaging raising concerns, since the start of 2023, there has been a moderate amount of research regarding this issue, detailing numerous methods to develop powerful models. Research from Schmedemann, et al. determined that using synthetic training data for defect detection on industrial surfaces was effective as they closely replicated actual surfaces[3]. They employed a generator that introduces defects as 3D models in the surfaces of objects, arguing that generating training data in this way is beneficial to the development of models when there is a lack of large amounts of annotated image data. In a separate paper, Banús, et al. discuss the relevance of using deep learning to find defects in food packaging specifically[4]. Using a pre-trained DesNet161 CNN, the results yielded a 0.03 to 0.30% range of false positives and a 0 to 0.07% range of false negatives., being able to detect 99.93% of sealing defects in any production.

Although this research is relevant for the field of quality control as a whole, there has been a lack of focus on specifically medicine package damage detection. Pharmaceutical packaging is at an all time high, making it very clear how important quality control for these packages is[5]. Leveraging the power of machine learning, it should be very possible to automatically find damages and irregularities on medicine package surfaces. The simple shape of most pharmaceutical packaging should make the research that we have done extremely applicable to most branded packages. The goal of our research is to step into the world of medicine package defect detection using artificial intelligence in order to optimize package quality control and to encourage more research to be done on this topic as this process is further optimized in terms of speed and accuracy.

Methods

The dataset chosen for this research was found on kaggle.com, uploaded under the name of Christian Vorhemus[6]. No work has been done on this dataset before. The dataset consists of a total of 400 540x960 procedurally created images of medicine packaging on a conveyor belt. Blender was the software used to create the synthetic machine learning training dataset[7]. It is the leading tool for realistic 3-dimensional image and animation generation which makes it appropriate for simulating real-life medicine packaging. As seen in Figure 1 below, one of the images generated for this dataset, the synthetic package is nearly

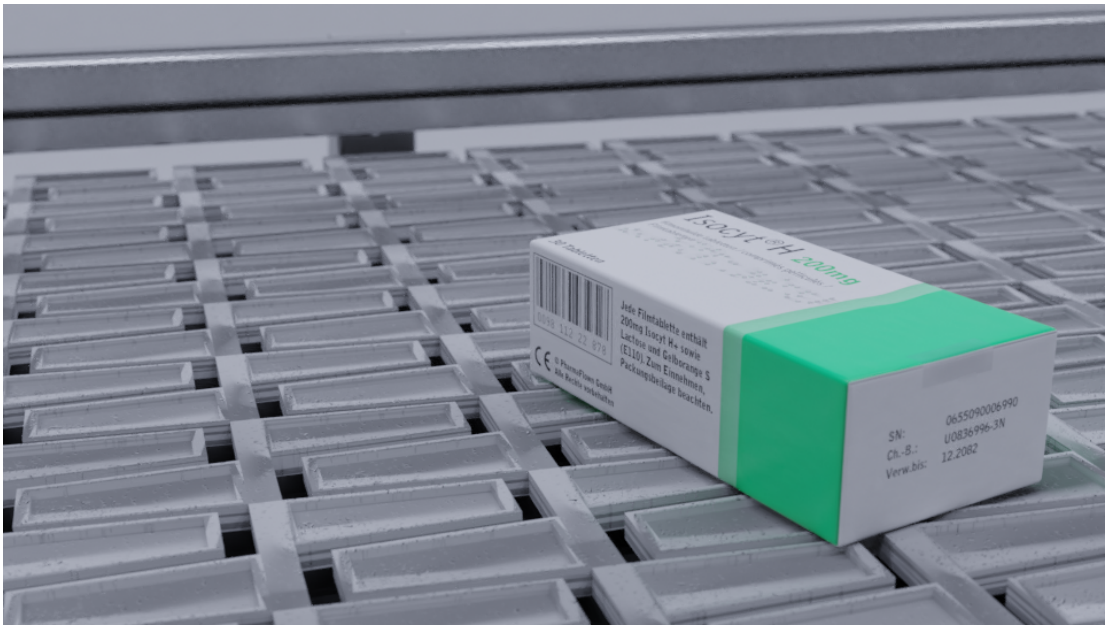

Fig. 1 A generated package image from the side view that resembles an actual package

identical to an actual pharmaceutical package.

Given the simplicity of these packages and the lack of variation between images, which will be emphasized through preprocessing techniques, we also determined that 400 images should be sufficient for training and testing. These images are separated into two classes being intact and damaged. Intact images depict packaging with minute or no visible damages to the exterior of the package. Damaged images depict packaging with significant damage to the exterior of the package. Both of these classes have two subclasses being side and top. Side images depict packages that are shown from a side view. Top images depict packages that are shown from a birds-eye-view. The result is that there are 100 images of intact packages viewed from the side, 100 images of intact packages viewed from the top, 100 images of damaged packages viewed from the side, and 100 images of damaged packages viewed from the top. The packages portrayed in the side and top class correspond to each other, meaning that the first image in the side view class and the second image in the top view class portray the same package. This results in a total of 100 intact packages and 100 damaged packages for the dataset. The data has a 75/25 split for training and testing data. The first 75 images of each subclass are reserved to train models while the last 25 images are reserved as testing data.

The initial goal of this research was to determine whether or not the package is damaged through using both perspectives of the package. In various cases, an image is detected as being damaged from only one of the perspectives, while the other perspective makes the package appear totally intact. Thus, both perspectives must be considered in order to make an accurate judgment. The most effective way to achieve this goal is to obtain a prediction for both perspectives of the package using two different models. If both models determine that the image is intact, then we also determine the image to depict an intact package. If either model, or both models determine that the image is damaged, then we determine the image to depict a damaged package.

A problem that arises with simply trying to process the data through a binary image classification CNN has to do with the nature of the data. The damages on the packages are often very subtle, and with such a small collection of images, a CNN has a difficult time extracting features to differentiate damaged and intact packages. Data augmentation was a technique we considered to solve the issue with the dataset size[8]. However, often the noise added by this method would interfere with the model's ability to determine the small damages, and there was a risk of overfitting to the augmented data if a significant amount of data was to be created. Transfer learning was another technique we attempted in order for a CNN to develop a deeper understanding of the data[9]. Learning from a preexisting ResNet50 model was applied to our dataset, however it did very little to increase the accuracy of our model[10]. We believe that more authentic data is necessary for a CNN to be applied.

Thus we determined that a deep learning approach is ineffective for this dataset as the features are simply too complex with the limited amount of data we have. As a result we resorted to using simple machine learning algorithms along with image processing techniques in order to develop a more successful model. We used four of the most common simple algorithms be-

ing Support Vector Machine (SVM), RandomForest (RF), Naive Bayes Classifier (NB), and K-nearest Neighbors (KNN)[11–14]. More advanced algorithms such as Gradient Boosting were considered, however with the limited processing power we had, they were unrealistic to run for many epochs[15].

Edge Detection

Since a model only needs information about the structure of the package, an edge detection algorithm that simplifies a colored image into an image of an image of white lines signifying an edge in the image is useful. Canny Edge Detection from the OpenCV library was found to yield a great result for our dataset[16,17]. Canny Edge Detection is an advanced algorithm that is extremely reliable in many contexts, making it a natural fit for our research. The way the algorithm works is by first applying a 5x5 Gaussian Filter to remove noise from the image[18]. Next, the image is filtered through a Sobel kernel, a more simple edge detection algorithm, in the horizontal and vertical direction[19]. Then, the image is fully scanned to remove unwanted pixels that do not constitute edges by checking at each pixel if it is a local maximum in its neighborhood in the gradient direction, resulting in a black and white image with "thin edges". Finally, a double thresholding technique is used to see which edges are true edges and which are not[20]. Values above the higher threshold are kept while values below the lower threshold are discarded. Values in between the thresholds are kept based on factors relating to the implementation. In this case, the thresholds indicate intensity gradient values for individual pixels, and values in between the thresholds are determined based on whether or not they are connected to pixels that have intensity gradients that are greater than the larger threshold. If a pixel with an in-between threshold value falls under that criteria, it is kept, while all other in-between pixels are turned off. At the end an image with strong edges is created. After experimenting with many values for thresholds, we decided to use a lower threshold of 100 and a higher threshold of 150 for our Canny Edge Detection. Figure 2 shows an original image and Figure 3 shows the image after edge detection for an intact and damaged package from the side.

Hyperparameter Optimization

To maximize the effectiveness of the algorithms that are going to be applied to the data we decided to optimize parameters for each algorithm. To do this, we applied the GridSearch tuning technique to all of the algorithms to determine the parameters that would yield the highest accuracy[21]. Given how similar the shapes and edges of each of these images are to each other, we are quite confident that aggressive tuning will not cause overfitting. In a real-world application, images should be captured in strict environments and thus result in little variation, which mirrors the appearance of our dataset.

Fig. 2 A package image from the side view without Canny edge detection

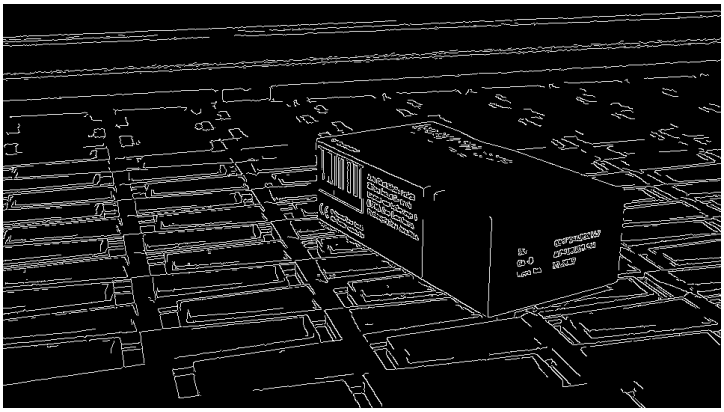

Fig. 3 A package image from the side view with Canny edge detection

Results

Through testing numerous values for parameters on GridSearch, we have decided to optimize our algorithms with parameters that result in the highest accuracy for each algorithm. GridSearch determined that a linear kernel and a C value of around 5 was most accurate for the SVM algorithm, having around 100 estimators with a max depth of around 10 was ideal for the RF classifier, a variable smoothing value of around 0.433 was best for the NB classifier, and having around 10 neighbors yielded highest accuracy for the KNN algorithm. These results were shared by both analyzing the top and side view separately.

After implementing Grid Search, the accuracies for each of the algorithms are listed in Table 1 without image preprocessing.

The accuracies for each of the algorithms after Canny edge detection are listed in Table 2.

The accuracy of these algorithms are relatively low, and can't

Table 1: Results for each model on the test data for each view without image preprocessing.

	Side View	Top View	Both Views
SVM	44%	44%	52%
RF	52%	52%	56%
NB	48%	48%	48%
KNN	56%	48%	52%

Table 2: Results for each model on the test data for each view with image preprocessing.

	Side View	Top View	Both Views
SVM	56%	52%	52%
RF	56%	52%	44%
NB	60%	40%	40%
KNN	56%	48%	48%

be applied to a real world context yet. We predict that the reason for this is due to how subtle some of the damages are. For example, for the top view damaged images shown in Figure 4 below, it is nearly impossible to determine that there is indeed a defect even with the human eye. Having a machine to pick up on the damage is even more difficult.

It appears that edge detection greatly helps with the model's ability to accurately determine if there were damages on the side view of the package. However, with the top view of the package, only certain algorithms are benefitted by preprocessing, while some are even hurt. As seen in Figure 3, the edges of the top view damaged medicine package don't seem to have defects on them, so edge detection overall didn't help the model much. For SVM, the side view and top view improved with edge detection. With RF, the side view accuracy improved but the top view accuracy was lower. For NB, a similar phenomenon occurred. Finally for KNN, the top view and side view accuracy was completely unchanged by image preprocessing. As seen in Figure 3, the edges of the top view damaged medicine package don't seem to have defects on them, so edge detection overall didn't help the model much. However, when considering both views of the package, the algorithms performed significantly worse after preprocessing. Given the way that the accuracy was determined, this means that the models were finding damages in the package at higher rates than they were supposed to, since we report a package as being damaged if either the side view or top view detect damages.

Using SVM as the model for the top view and NB as the model for the side view after preprocessing, the overall accuracy ended up being 48%. This indicates that there are a significant amount of false positive scores. By looking into the predictions themselves. We found that the SVM model detected 24 out of the 25 packages in the test set as being defective. This contrasts with how the NB model detected only around half of the packages as being defective.

We believe that the conveyor belt in the background of the images may be a source of these false positives. The jagged lines may confuse some models after applying edge detection. The irregularity of the lines may have resulted in models more sensitive to positive results. Some algorithms may be more susceptible to this than others, however more research needs to be done to make a definitive conclusion.

Discussion

Out of each of these models, the RF classifier on the unaltered images had the highest accuracy when we considered both views of the package. It was the only model to have consistently achieved an accuracy for the side view, top view, and overall package of over 50%.

Canny edge detection seems to be a useful tool for finding damages on the side view specifically. However, it also seems to create false positives, and overall, it is harmful for overall accuracy, at least for these basic algorithms.

The accuracy of these algorithms was relatively low for this dataset, and we attempted to implement many preprocessing and recognition techniques, many of which failed. Using a CNN was our first idea, however we couldn't improve the accuracy from 50% no matter how many factors of the CNN we changed. We determined that the quantity and complexity of the data was the limiting factor, however there may very possibly be a way to use CNN's to get stronger results if the dataset was expanded or better processed. Data augmentation, as we mentioned before, was considered, but was disregarded due to the homogeneity of the images we had to work with.

We also tried to implement object detection and foreground extraction to simplify the images, however they all came with severe drawbacks. A MediaPipe object detection framework was originally considered for this dataset, however, it failed to detect objects for around 25% of the images taken from the side view[22]. A foreground extraction method called GrabCut from OpenCV was also considered, but it often failed to remove the background neatly from the side view images[23]. A next step would be to implement our own background removal engine that is catered to this dataset specifically. We believe that the noise from the conveyor belt of the images will be removed this way, and once combined with edge detection, can feed the most concise and necessary information to the model.

Fig. 4 A package image from the top view that is damaged despite looking intact.

Conclusion

Although the pharmaceutical packaging industry can't take immediate advantage of the methods proposed in this research, our findings demonstrate the limiting factors that future researchers can expect to face. With the use of larger datasets that contain images that are easier to interpret by a machine, we believe significant progress in the success of image recognition in this field can be made. We hope to inspire future work in medicine packaging quality control and warn of potential roadblocks for various machine learning algorithms.

Acknowledgements

Thank you for the guidance of Deepak Badarinath from Oxford University and Chien Erh Lin from the University of Michigan in the development of this research paper.

References

1 G. Bodenheimer, *Packaging Digest*, 2019.

2 Y. Jiang, *Amcor*, 2020.

3 O. Schmedemann, M. Baaß, D. Schoepflin and T. Schüppstuhl, *ScienceDirect*, 2022, **107**, 1101–1106.

4 N. Banús, I. Boada and P. Xiberta, *Scientific Reports*, 2021, **11**, 21887.

5 *Contract Pharma*, 2024.

6 C. Voerhemus, *Kaggle*, 2020.

7 *Documentation*, `https://docs.blender.org/`.

8 C. Dilmegani, *AIMultiple*, 2023.

9 N. Donges, *Built In*, 2022.

10 *resnet50*, `https://www.mathworks.com/help/deeplearning/ref/resnet50.html`.

11 R. Gandhi, *Towards Data Science, Medium*, 2018.

12 T. Yiu, *Towards Data Science, Medium*, 2019.

13 R. Gandhi, *Towards Data Science, Medium*, 2018.

14 O. Harrison, *Towards Data Science, Medium*, 2018.

15 R. Joseph, *Towards Data Science, Medium*, 2018.

16 *Canny Edge Detection*, `https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html`.

17 *OpenCV*, `https://opencv.org/`.

18 *Gaussian Filtering*, 2005, `https://www.southampton.ac.uk/~msn/book/new_demo/gaussian/`.

19 R. Fisher, S. Perkins, A. Walker and E. Wolfart, *Sobel Edge Detector*, 2000, `https://homepages.inf.ed.ac.uk/rbf/HIPR2/sales.htm`.

20 *Double Threshold*, 2023, `https://support.echoview.com/WebHelp/Windows_And_Dialog_Boxes/Dialog_Boxes/Variable_Properties_Dialog_Box/Operator_Pages/Double_Threshold.htm`.

21 *Hyperparameter Tuning with GridSearchCV*, 2023, `https://www.mygreatlearning.com/blog/gridsearchcv/`.

22 Kukil, *Introduction to MediaPipe*, 2022, `https://learnopencv.com/introduction-to-mediapipe/`.

23 Ray, *Analytics Vidhya, Medium*, 2020.